COMP 47330
DINNER BUDDY

Terry McCartan

13206072

Table of Contents

Application Overview

Introduction

Dinner buddy is a recipe suggestion app that leverages on the Yummly API. The app takes a list of ingredients and suggests meals that can be cooked using them. The user can view a summary of the recipe and can view the recipe on the reference site for detailed steps. The user can save recipes that they have searched for before for quick reference.

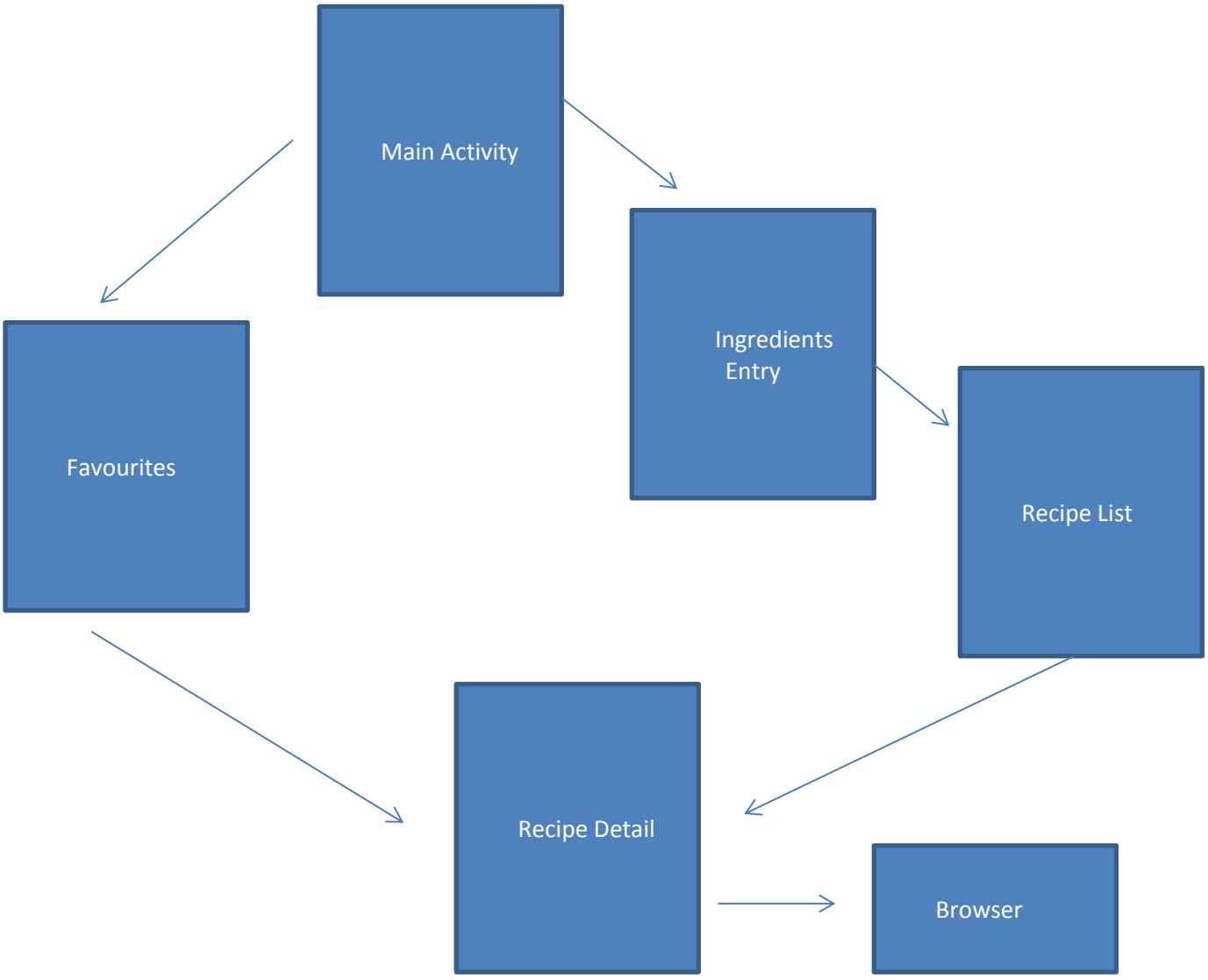
High level functionality

On load of the application the user is brought to the main screen. From here they can either select their recipe favourites or enter in ingredients for searching. The ingredients activity allows for the user to enter in ingredients in free text. The user can see a list of entered ingredients and remove them from the list. When the user enters in ingredients and clicks on the search button they are brought to the recipe list activity. This is a custom list view that provides the user with the recipe name and a small image. The user can select a recipe and be brought to a detailed page describing more details about the recipe. From this activity the user can return to the list, save the recipe to their favourites or view the recipe on the source site for more details.

In the favourites activity the user can

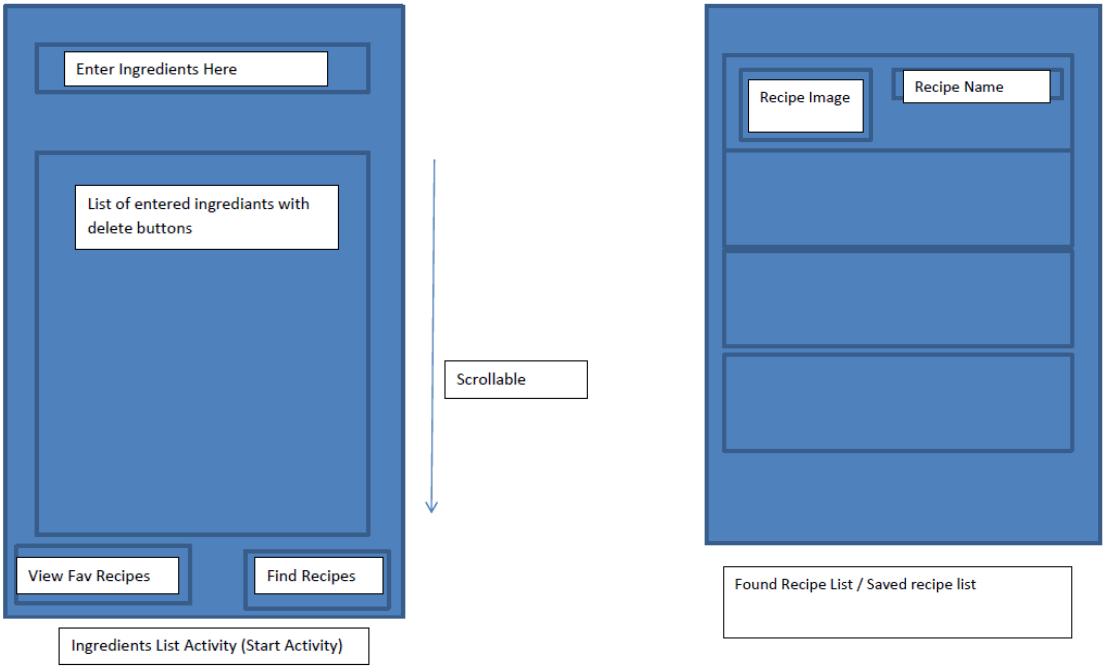
App Flow

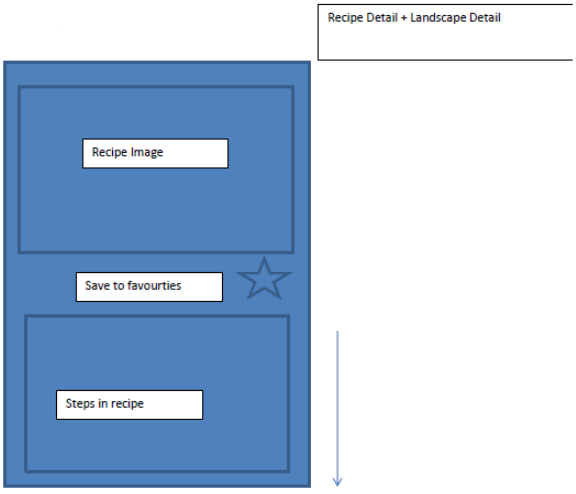
A map of the activity flow is shown below

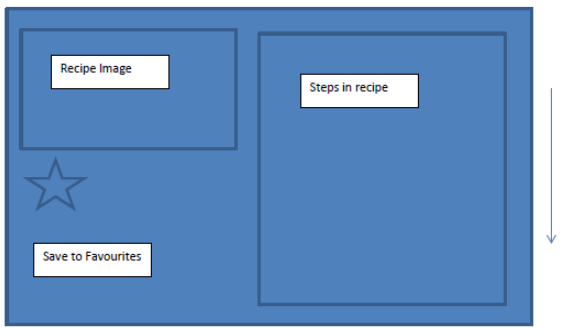


Mock up/s

Initial mock ups were made at the project proposal stage. These mock ups were wireframes which acted as placeholder for where I wanted functionality to be. These mock-ups helped plan the activity flow and at the implementation stage some holes were found with the initial mock-ups.

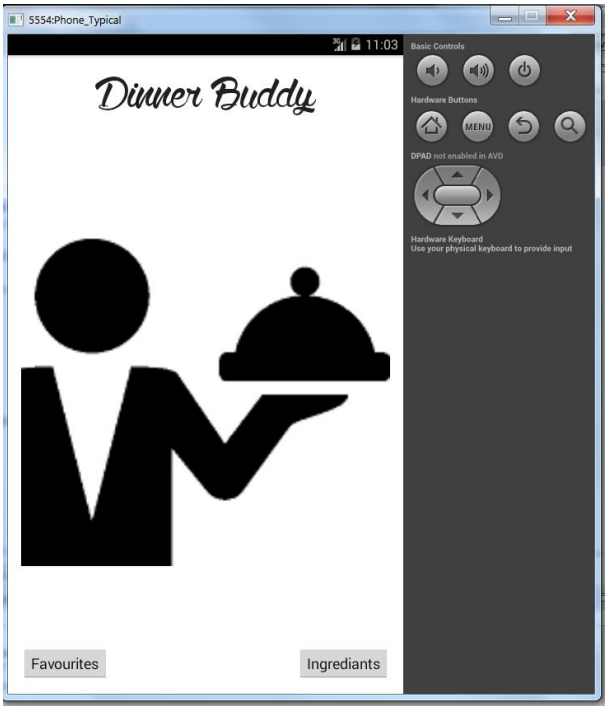






Detailed breakdown

Main Activity



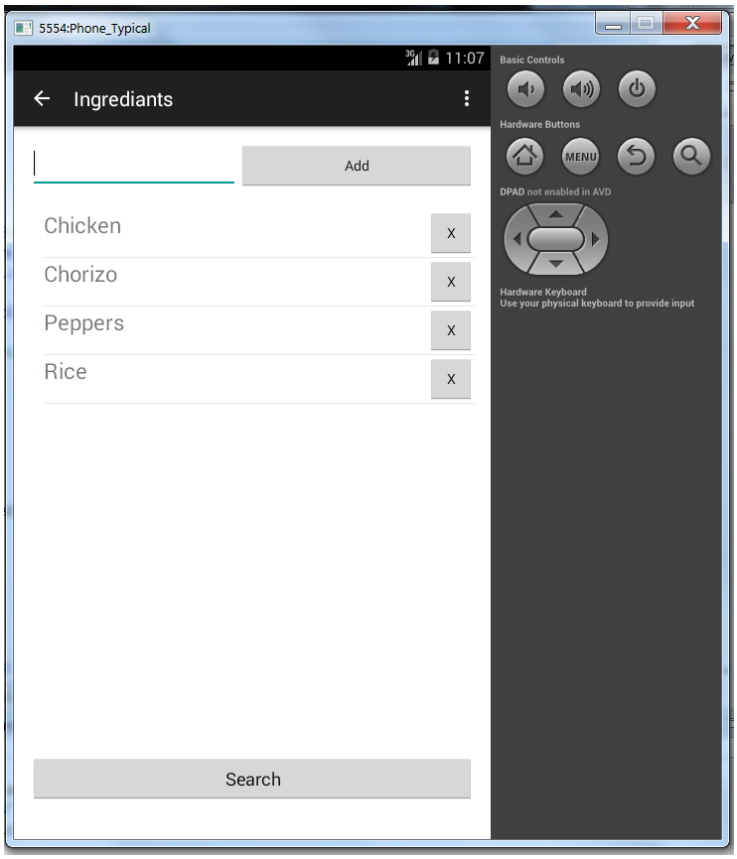
The launcher activity for dinner buddy is Main Activity. From this screen the user can access the favourites list and the Ingredients entry activity. This screen is comprised on a text view, an image view and two buttons. The TextView has a custom font applied to it which was made available be a .ttf file that can be accessed here [4]. More details about this can be found later in the document in the Technologies sections.

The image serves as a logo for the app and has been scaled down to suit individual screen sizes. The image resolution is the following

- Ldpi 192x192
- Mdpi 192x192
- Hdpi 384x384
- XHdpi 512x512
- XXHdpi 512x512

Some initial difficulties with the design of this activity were making the image fit correctly on each device. This was due to using the same image resolution instead of breaking them down into the individual folders and reducing the size of the image. I used gimp to scale down the original image and placed them in the correct directory following guidelines from the android developer API [12] another issue was the use of the custom font. Custom fonts aren't supported in the graphical designer so text size was difficult to judge. This was very much a trial and error fix were eventually I found a number that suited the font and displayed it the way I wanted.

Ingredients List Activity



The ingredient activity provides the user with the ability to enter in specific ingredients that will be used to search through the Yummly API. The user can enter in free text and add the ingredients to the list. The user can remove items from the list by clicking on the X button. Once the user has entered in the desired ingredients they can use the search button to proceed to the recipe list activity. The ingredients list is persisted into Shared Preferences so that if the app is stopped or the user goes out of the app, the list is persisted.

One of the complications in developing this activity was the addition of dynamic items to the list view and removing the items from the list view. To handle the addition of items to the list, the adapter of the list view exposes a method called notifyDataSetChanged. This handles any addition or removal of items from the list.

To handle the second issue a button was added into the list view item template that acts like a delete button. The button's click event was attached onto in the adapter getView method. When clicked, it would remove the item from the adapters list and again call notifyDataSetChanged.

The final issue was the persisting of the items into shared preferences. Since the items are deleted in the adapter, it did not seem like a good place to delete them from shared preferences. I decided that the best solution was to override the OnDestroy method of the activity and replace the items that were in shared preferences with current updated items. On create the items are taken from the shared preferences and the list view is updated correctly.

Yummly Controller + JSON

Async tasks

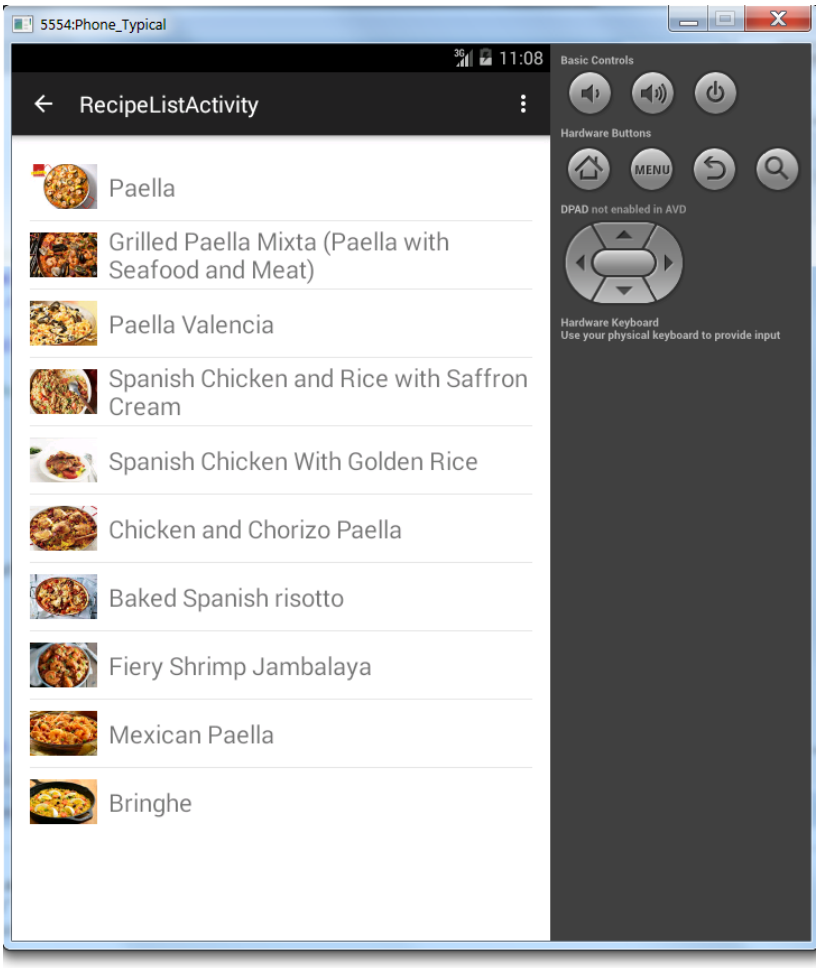
In developing the object that would access the Yummly API, I came across an issue. This issue was to do with accessing the API over the network on the main UI thread. After reading about what the best practice when accessing HTTP Rest API, I decided to create an Async task object that would handle the HTTP request. Once the task was completed it would callback into the calling class with the received data. More information about accessing the Yummly API is found in the technology section. The yummly requests are only setup to take the first ten results from the API search requests. The reason for this is explained in future work section.

```
public class DownloadRecipeAPIAsyc extends AsyncTask <String, Integer, String>{

	@Override
	protected String doInBackground(String... params) {
		HttpGet httpGet = new HttpGet(params[0]);
		ResponseHandler<String> handler = new BasicResponseHandler();
		String response = "";
		try {
			response = _client.execute(httpGet,handler);
		} catch (ClientProtocolException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}
		return response;
	}
	protected void onPostExecute(String result) {
	  GetRecipeCompleted(result);
	}

}
```

Parsing JSON

Once the information is retrieved from the API, the data needed to be parsed into an object that can be used throughout the app. This led to the creation of the Recipe object which models the return from the API methods. Parsing out the JSON string into the Recipe object required using the JSONObject and JSONArray classes from the org.json namespace. The different API calls (Search/Get) exposes varying levels of detail to with the recipes so parsing the information was difficult at the beginning.

Recipe List Activity

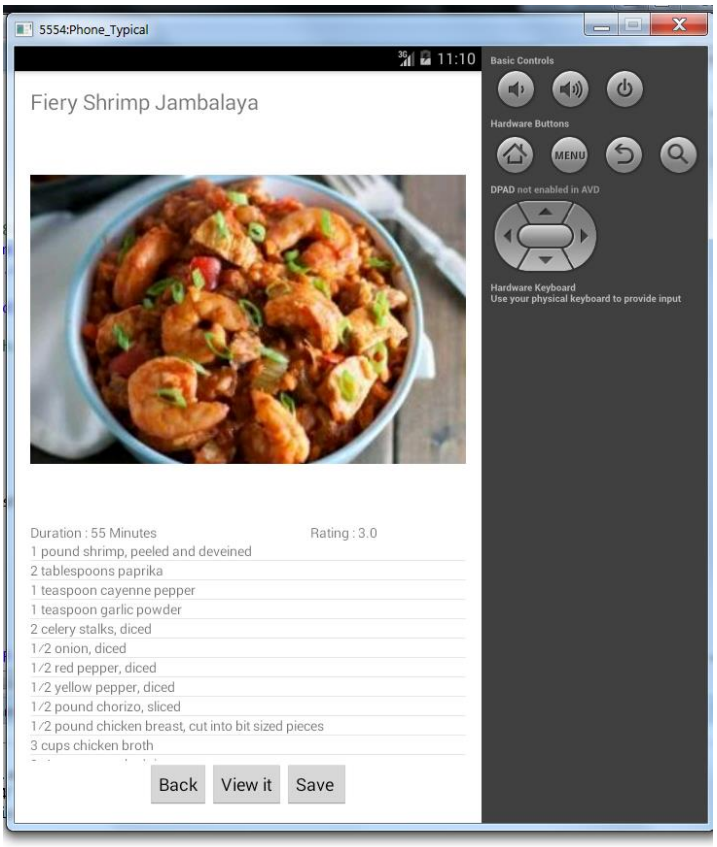


This activities responsibility is to display recipe search results. The activities intent is passed the search results from the previous activity. The activity calls the JSON parser to parse out the results and passes the list to a custom data adapter. The user can select a recipe to view more information about it.

The activity will add its recipe search results to shared preferences so that when the activity is stopped or interrupted by the user, they can continue on where they left off rather than search for the recipes again.

One of the issues when creating this activity was showing the image thumbnails in the list view. I originally went with downloading the image directly using the Bitmap Factory with an URL.openStream. This caused issues with executing network requests on the main thread which led to me creating the ImageDownloadAsync task class. I originally had trouble with getting reference back to the Image view to apply the bitmap to. After a bit of research I decided to follow the view holder pattern which held reference to the image view [view holder reference].

Recipe Detail Activity

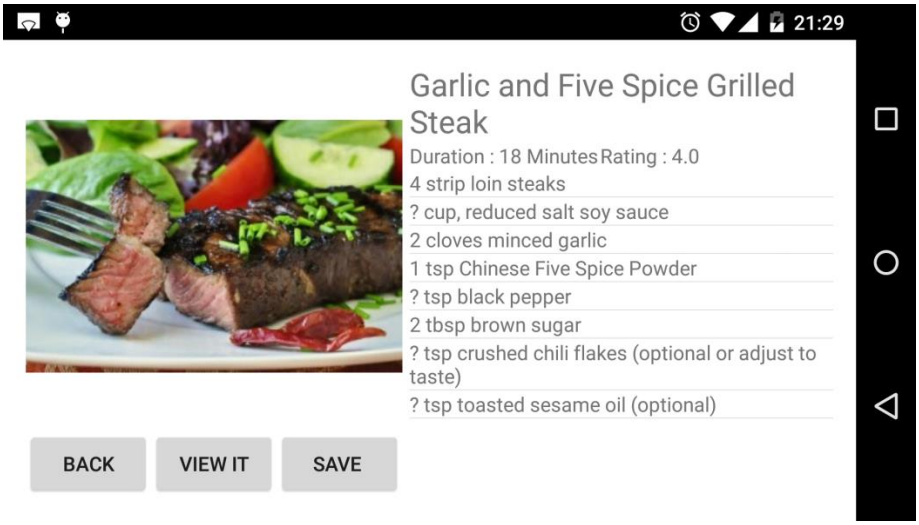


The recipe detail provides the user with more information about the selected recipe. The activity starts by accessing the Yummly API and getting the detailed recipe information. This can be a slow process depending on the user's network connectivity and Yummly responsiveness. To mitigate this issue a progress bar is added to the activity as an indeterminate busy indicator.

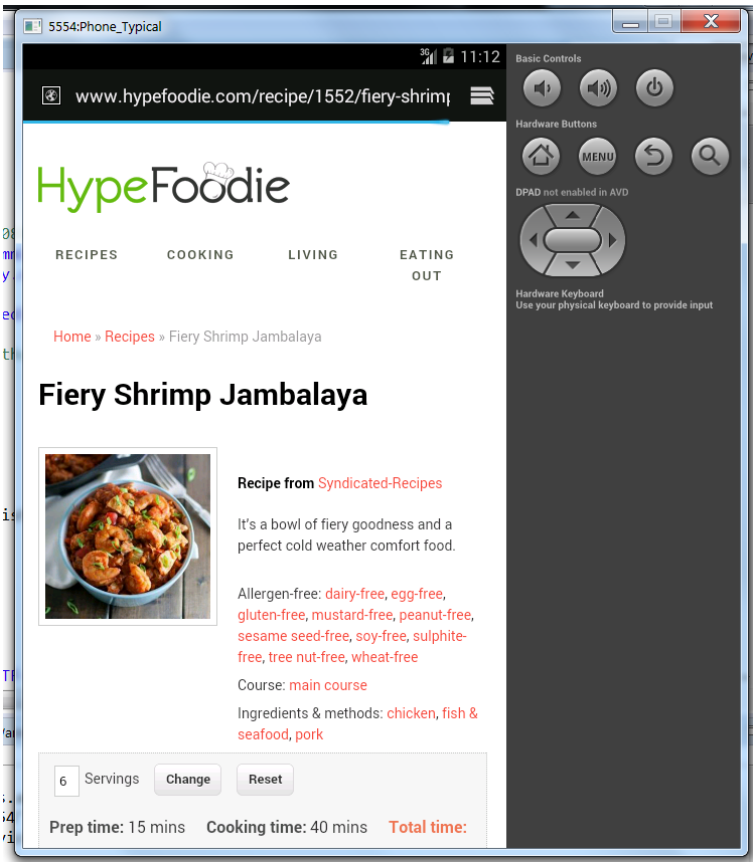
An issue with design this view was that the ingredients list for each recipe can be different length so just a standard text layout was not possible. I got around this by again using a list view that will scroll over the given ingredients

If the user desires so, they can view the recipe on the source website. This opens a browser activity with the URL pre-set. I had issue with this, since the browser intent needed the HTTP scheme in the URL provided. This was handled by the API since the source URL provided is always absolute.

As an added style decision, I decided to remove the activity bar from the activity by inheriting from the Activity class rather that the ActivityBar activity. I thought this gave more intention toward the content of the activity. This decision meant that I needed to include a back button, which I added to the bottom button bar



The recipe detail activity has been optimised for horizontal viewing by adding a new xml file into the app. To support the horizontal screen a new folder was added that was named layout-land. This naming convention allows the OS to load the landscape xml once the screen is rotated.



The above screenshot is a sample recipe that was viewed from the app. The user can view more detail about the activity that would be available through the app. If the user wants to, the can use the back button to return back to the application, on the recipe detail page.

Favourites Activity

The purpose of the favourite's activity is to allow the user to save recipes that they have seen before for quick access. This allows the user to maintain a list of recipes that can have various ingredients. The design of this activity closely resembles that of the recipe list and therefore I was able to reuse the xml file from that class to display the favourites list. This allowed me to rapidly put together the favourite's activity, but I am unsure if this approach is a sustainable solution. It very much couples the two activities together so making a change in one can seriously affect the other. Nonetheless the approach worked for my specific case.

The favourite's functionality makes use of the Shared Preference functionality in android. Much like the ingredients list, the favourites are placed into the shared preferences and stored for later use. The main difference is that the favourites are objects rather than strings so I decided to use GSON [3] to handle the serialization of objects into the shared preferences and de-serialise them back into objects when needed. The reason I chose to do it this way instead of using the setSet functionality of share preferences was to maintain API support of 10 (setSet was introduced in 11)

I decided to use the shared preference as a storage mechanism because of familiarity and its lightweight design. Instead of using a traditional SQLite storage mechanism, the shared preferences offed a key value storage solution that was perfect for my needs. From investigation, there doesn't seem to be a hard limit on how high the shared preferences can go (device dependant) and the preferences allows the storage of object even when the application is killed and restarted.

To provide the user with the favourite's functionality, I placed a button on the recipe detail page that will allow the user to save it to their preferences. Once the item was added to the preferences the button would change its text to "Remove". Clicking on this button will then remove the specific recipe from the preferences. When the Recipe detail page loads, it always checks if the loaded recipe is already in favourites and will set the button appropriately.

The favourites list allows the user to open back up the recipe detail activity to view a summary about the recipe and provides the user with the ability to view the recipe in more detail. The ability to remove from favourites is also provided in this activity so the user can quickly remove recipes from their favourites. I found this a very useful way of tying the application together where the user is familiar with the screens they seen and they do not require a lot of training/use to get around.

Technologies

Yummly API +Samples

The primary 3 party technology I use is the Yummly API. Yummly is a recipe aggregator site that compiles list of recipes that can be view on their website. Their API is not free but I was able to obtain an academic licence for the purpose of this project. The API provides two main REST methods for accessing the recipes.

Search

The recipe search method allows applications to search for recipes by providing parameters that return filtered results. Dinner Buddy leverages on this functionality, specifically the allowed ingredients parameters. A sample Http get string is proved below

```
http://api.yummly.com/v1/api/recipes?_app_id=id&_app_key_=key&allowedIngrediant=chicken
```

This request returns a JSON result of the recipes, which dinner buddy parses and displays to the user. A sample output from this request is below


```
{"attribution":{"html":"Recipe search powered by <a href='http://www.yummly.com/recipes'><img
alt='Yummly' src='http://static.yummly.com/api-
logo.png'/></a>","url":"http://www.yummly.com/recipes/","text":"Recipe search powered by
Yummly","logo":"http://static.yummly.com/api-
logo.png"},"totalMatchCount":337605,"facetCounts":{},"matches":[{"imageUrlsBySize":{"90":"http://lh3.goo
gleusercontent.com/5nPdEmY7ppS_uN85LGE0JSKzV_zgCqANfroPUAtMJFbQ2SwWVo8ff6BWXKEDyrQzribBHDIUo79e89wJoiRjP
cs=s90-c"},"sourceDisplayName":"The Country Contessa","ingredients":["elbow macaroni","butter","all-
purpose flour","milk","salt","shredded cheese"],"id":"Baked-Macaroni-and-Cheese-
1083868","smallImageUrls":["http://lh3.googleusercontent.com/7LwU_6KPHdlG8rqoqKwMYbkfz8b2VSp386dznquGmFz
6XC8E8gcWKAPQ0KPiXavj05u3oAW0_K_fYsPZit-B8mU=s90"],"recipeName":"Baked Macaroni and
Cheese","totalTimeInSeconds":2400,"attributes":{"course":[],"cuisine":[]},"flavors":{"salty":0.666666666
6666666,"sour":0.1666666666666666,"sweet":0.1666666666666666,"bitter":0.8333333333333334,"meaty":0.666
6666666666666,"piquant":0.0},"rating":4},{"imageUrlsBySize":{"90":"http://lh3.googleusercontent.com/THc5
X3L9geQS-oEcxbmfqqoixngl4uJSQrQIALsr3WDSiHAbCiWdqRHv23dZnuvXejZogrfiotggxQ8GabBlQQ=s90-
c"},"sourceDisplayName":"Just a Taste","ingredients":["rolled oats","milk","chia seeds","agave
nectar","fruit"],"id":"Healthy-Overnight-Oats-with-Chia-
1084502","smallImageUrls":["http://lh3.googleusercontent.com/rPBLObR5_TMGTRk6_37WDFTaXQW8DwrQB6FDQn61Hnq
YsMOUpbsDK250OBSUpxPnR0jT3AarHOumxSgu7wWbYcI=s90"],"recipeName":"Healthy Overnight Oats with
Chia","totalTimeInSeconds":43200,"attributes":{"course":["Breakfast and
Brunch"]},"flavors":null,"rating":4},{"imageUrlsBySize":{"90":"http://lh3.googleusercontent.com/BWmGrX2n
k44QqKtM64YWS9khNHDy4mutUF8RMLWx8NODcYeQ_aTs4a6pI85aPFeydthv3dbLdn8TtmkAx1QKeQ=s90-
c"},"sourceDisplayName":"Enjoy Tribute","ingredients":["tomatoes","onions","red bell pepper","olive
oil","bouillon","sea salt"],"id":"Three-Ingredient-Tomato-Soup-
1082523","smallImageUrls":["http://lh3.googleusercontent.com/tbgH44_aFVRT4izSEI2Phqrh0mL0IOQRW2RpmA_1Fva
QWgPGSNNC3ZjLTbH9z1CvLIYVJ9jV798s82uhYWBmaA=s90"],"recipeName":"Three Ingredient Tomato
Soup","totalTimeInSeconds":1800,"attributes":{"course":["Soups"]},"flavors":null,"rating":4},{"imageUrls
BySize":{"90":"http://lh3.googleusercontent.com/3tddQoDCON_ebgGnn-l8mjrrsMKLAIG99taDRTFCcCwroaEgKP64-
eRq1VUAWiOcQpnWXYuSXq9qn6-9hntrog=s90-c"},"sourceDisplayName":"Twin Dragonfly","ingredients":["beef
stock","onion soup mix","roast","diced onions"],"id":"Easy-Slow-Cooker-Beef-Dip-
1083579","smallImageUrls":["http://lh3.googleusercontent.com/w3TjxyZUlaIhJc7FZBNbL2lsU1FaDgDmSWbLH5QrvjF
Fzq0I926Z7TtwH6fSkIkJ0YRXSSqz93-AHHwsQZtp=s90"],"recipeName":"Easy Slow Cooker Beef
Dip","totalTimeInSeconds":29100,"attributes":{"course":["Appetizers"]},"flavors":null,"rating":4},{"imag
eUrlsBySize":{"90":"http://lh3.googleusercontent.com/Rg3iMDfFJqVDsomg_jRBvL1BPc9Yi5AtNmPyWZ4EP5NNR7Z2JK_
_8_fAT4S-_JO91H5UgRBSTyGakfUPkfQvFdI=s90-c"},"sourceDisplayName":"Further
Food","ingredients":["spaghetti squash","thyme","olive oil","butter","sage
leaves","salt","pepper"],"id":"Spaghetti-Squash-with-Sage-Brown-Butter-
1082308","smallImageUrls":["http://lh3.googleusercontent.com/nQJniYK0WlRcIl881J5-
```


The search functionality provide the user with high level information about the recipes, for a more detailed view of the recipe, the application must make use of the API get recipe method.

Get

The API provides a mechanism to get more detailed information about a recipe. The get recipe exposes a recipe id that is returned from the search method. To get the detailed recipe information the App just needs to send the recipe id as a parameter. An example HTTP request is shown below

http://api.yummly.com/v1/api/recipe/Honey-Dijon-Garlic-Chicken-Breasts-1072650?_app_id=9025e66c&_app_key=fc7f68e152a7d3e088cd75f5da008cb5

A sample output of a detailed recipe request is shown below. (Note for clarity, the nutritional information returned is omitted as it is quite large amount of text).


```
{"yield":"4","nutritionEstimates":[….],"totalTime":"40
min","images":[{"imageUrlsBySize":{"90":"http://lh3.googleusercontent.com/tooUczX87NE9PfvsUOG19kTL
_hrVoFIVnlZzG9HA5GmyZa7lb_e5jkmeVcqeuUcih68FpzxGXzL7bX_-4KrjFw=s90-
c","360":"http://lh3.googleusercontent.com/tooUczX87NE9PfvsUOG19kTL_hrVoFIVnlZzG9HA5GmyZa7lb_e5jkm
eVcqeuUcih68FpzxGXzL7bX_-4KrjFw=s360-
c"},"hostedSmallUrl":"http://lh3.googleusercontent.com/FtRRydQRhRKXDG6E_cFrJ9Gvrpsy0yn-
H0tiXzSKsn_ck8IZbzbfW7z46bO_MEorQdrhxekJmuAGYcNaEQLA7w=s90","hostedMediumUrl":"http://lh3.googleus
ercontent.com/FtRRydQRhRKXDG6E_cFrJ9Gvrpsy0yn-
H0tiXzSKsn_ck8IZbzbfW7z46bO_MEorQdrhxekJmuAGYcNaEQLA7w=s180","hostedLargeUrl":"http://lh3.googleus
ercontent.com/FtRRydQRhRKXDG6E_cFrJ9Gvrpsy0yn-
H0tiXzSKsn_ck8IZbzbfW7z46bO_MEorQdrhxekJmuAGYcNaEQLA7w=s360"}],"name":"Honey Dijon Garlic Chicken
Breasts","source":{"sourceRecipeUrl":"http://www.rockrecipes.com/honey-dijon-garlic-chicken-
breasts/","sourceSiteUrl":"http://www.rockrecipes.com","sourceDisplayName":"Rock
Recipes"},"id":"Honey-Dijon-Garlic-Chicken-Breasts-1072650","ingredientLines":["4 large boneless
skinless chicken breasts, about 6 ounces each","3 tbsp butter","6 cloves minced garlic","pinch
salt and pepper","? cup honey","2 tbsp whole grain Dijon mustard"],"attribution":{"html":"<a
href='http://www.yummly.com/recipe/Honey-Dijon-Garlic-Chicken-Breasts-1072650'>Honey Dijon Garlic
Chicken Breasts recipe</a> information powered by <img alt='Yummly'
src='http://static.yummly.com/api-logo.png'/>","url":"http://www.yummly.com/recipe/Honey-Dijon-
Garlic-Chicken-Breasts-1072650","text":"Honey Dijon Garlic Chicken Breasts recipes: information
powered by Yummly","logo":"http://static.yummly.com/api-
logo.png"},"numberOfServings":4,"totalTimeInSeconds":2400,"attributes":{"course":["Main
Dishes"]},"flavors":{"Piquant":0.1667,"Meaty":0.8333,"Bitter":0.5000,"Sweet":0.6667,"Sour":0.1667,
"Salty":0.5000},"rating":4}
```


Android Technologies

This section provides a high level view of some of the android specific technologies I used in the process of creating Dinner Buddy.

- Shared preferences
  I used shared preferences to store information that has been entered into application and to persist states of activities when the app is stopped/killed. The shared preference key value style store allowed for me to easily add and remove items without adding unneeded complexity.

- Custom Adapters
  For displaying the list of recipes and ingredients I used a list view. To use the list view however I needed to implement custom adapters that would display the items in a specified template. For the ingredients it was easy to add a row template with a remove button that when called could delete the item from the list and update the UI. For the recipes it was a bit difficult to implement the list view with the url image, but after some research and the implementation of the download image URL task it became simple.

- Images + Async tasks
  To display the recipe images that the API provided images needed to be downloaded by URL. The initial solution was to download the image on the main thread, but I ran into network exceptions. A bit of research led me to async tasks, but I ran into issues getting the downloaded image back into the ImageView. I came across the ViewHolder Pattern [5] and implemented it to keep reference to the image view.
- JSON Parsers
  Due to the decision of using the shared preferences, I needed ability to convert the object to string and back again. This led me to look at the native serialisation of android which I found pretty much non-existing. To get around this issue, I decided to look at third party jars. This led me to the GSON project from google [3] which suited my needs perfectly.
- Custom fonts
  As an added feature, I wanted the logo screen to have a nice font to show off the application. I found the built in fonts ok but not suitable for my needs. Instead of creating the text in Photoshop or gimp, I decided to look at custom font solutions. I came across a .tff file that was located here [4]. Adding this font was a bit tricky, since android does not allow custom fonts in the graphical designer. To added the font I needed to use the following method

```
Typeface font = Typeface.createFromAsset(getAssets(), "custom_font.ttf");
txtMain.setTypeface(font);
```

Conclusions

Future Work

There are areas that I would like to add into App. One of the areas is the social aspect of mobile computing. I'd like to integrate social media logins and have the ability to share or send recipes to each user. This would be a great add on as it would allow users to find recipes that others suggest with ingredients they may not be familiar with.

Another area would be removing the browser aspect of the App by including the ingredients into the recipe detail. This would require changing out the Yummly API into one that provides the ingredients in its response. Another option is to create recipes within the App and remove the dependency on third part APIs. Removing the API would lead to other areas of the application to improve and for new areas to be implemented.

The other areas I would like to improve on is UI and yummly search results. For the UI, I'd like to support smaller screens better, I had trouble with the ImageView control on smaller devices but due to time constraints I had to move on. The final area I'd like to improve on is change the recipe list activity to request the next set of search recipes when the user hits the bottom of the list view. Again due to time constraints I had to omit this behaviour.

Final Word

I felt the development of the application increased my ability to program within the android environment. The interaction with the REST API allowed me to gain skills that are used in enterprise when mobile applications interact with larger systems. The design of the layouts and the work done on the previous assignments allowed me access to get familiar with the UI design of the SDK. In closing, I found designing and implementing the App to be a worthwhile and rewarding experience.

Bibliography

1. Activity, P. (2015). *Progress & Activity | Android Developers*. [online] Developer.android.com. Available at: http://developer.android.com/design/building-blocks/progress.html [Accessed 22 Apr. 2015].

2. Androidhive.info, (2015). *Android JSON Parsing Tutorial*. [online] Available at: http://www.androidhive.info/2012/01/android-json-parsing-tutorial/ [Accessed 22 Apr. 2015].

3. Code.google.com, (2015). *google-gson - A Java library to convert JSON to Java objects and vice-versa - Google Project Hosting*. [online] Available at: https://code.google.com/p/google-gson/ [Accessed 22 Apr. 2015].

4. Dafont.com, (2015). *Wolf in the City Font | dafont.com*. [online] Available at: http://www.dafont.com/wolf-in-the-city.font [Accessed 22 Apr. 2015].

5. Developer.android.com, (2015). *AsyncTask | Android Developers*. [online] Available at: http://developer.android.com/reference/android/os/AsyncTask.html [Accessed 22 Apr. 2015].

6. Developer.yummly.com, (2015). *Yummly | Recipe API & Food API*. [online] Available at: https://developer.yummly.com/ [Accessed 22 Apr. 2015].

7. Options, S. (2015). *Storage Options | Android Developers*. [online] Developer.android.com. Available at: http://developer.android.com/guide/topics/data/data-storage.html#filesInternal [Accessed 22 Apr. 2015].

8. Stackoverflow.com, (2015). *Android - Using Custom Font*. [online] Available at: http://stackoverflow.com/questions/3651086/android-using-custom-font [Accessed 22 Apr. 2015].

9. Stackoverflow.com, (2015). *Load image from url*. [online] Available at: http://stackoverflow.com/questions/5776851/load-image-from-url [Accessed 22 Apr. 2015].

10. Stackoverflow.com, (2015). *Sending POST data in Android*. [online] Available at: http://stackoverflow.com/questions/2938502/sending-post-data-in-android [Accessed 22 Apr. 2015].

11. Tutorialspoint.com, (2015). *Android Loading Spinner Tutorial*. [online] Available at: http://www.tutorialspoint.com/android/android_loading_spinner.htm [Accessed 22 Apr. 2015].

12. Screens, S. (2015). *Supporting Multiple Screens | Android Developers*. [online] Developer.android.com. Available at: http://developer.android.com/guide/practices/screens_support.html [Accessed 22 Apr. 2015].

13. Developer.android.com, (2015). *Making ListView Scrolling Smooth | Android Developers*. [online] Available at: http://developer.android.com/training/improving-layouts/smooth-scrolling.html [Accessed 22 Apr. 2015].